Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Bachelorarbeit (Informatik)

Unity 3D Visualisierung für Orbitarium

Autoren	Kevin Thalmann Tim Krenz
Hauptbetreuung	Karl Rege

Datum	11.06.2021

Erklärung betreffend das selbstständige Verfassen einer Bachelorarbeit an der School of Engineering

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:	**Name Studierende:**
Hausen AG, 11.06.2021	Kevin Thalmann
Birmensdorf ZH, 11.06.2021	Tim Krenz

Zusammenfassung

Im Jahr 2020 wurde im Rahmen einer Bachelorarbeit eine Software-Suite erstellt, um ein Orbitarium zu betreiben. Diese Software-Suite ist eine Sammlung verschiedener Programme, die benötigt werden, um das Orbitarium zu betreiben. Folgend wird sie Orbitarium-Suite genannt. Das Orbitarium selbst ist eine Plexiglaskugel, auf welche mit Projektoren die Oberfläche der Erde projiziert wird, um verschiedene Animationen darauf abzubilden. Es gehörte ehemals dem Technorama Winterthur und wurde ebenfalls im Jahr 2020 von der ZHAW übernommen. Aktuell steht das Orbitarium in einem provisorischen Raum. In Zukunft soll es auf dem ZHAW Areal ausgestellt werden.

Die Version der Orbitarium-Suite aus der Vorarbeit hat ein Performanceproblem. Abgespielte Animationen werden mit ca. 3 Bildern pro Sekunde dargestellt und ruckeln dadurch stark. Durch einige Anpassungen im Code und durch den Austausch eines Programms der Suite, wird die Performance auf ca. 15 Bilder pro Sekunde erhöht. Es wird eine Klimasimulation implementiert, welche auf einer Masterarbeit aus dem Jahre 2010 basiert und später von Herrn Rege in einer Java Applikation implementiert wurde. In dieser Bachelorarbeit wird die Java Applikation in die Orbitarium-Suite migriert, um sie auf einem echten Globus anstelle eines virtuellen abzuspielen. Die Orbitarium-Suite wird um die Funktionen erweitert, Animationen aus Videos und Webseiten anzuzeigen. Diese neuen Möglichkeiten erleichtern das Anzeigen neuer Inhalte im Vergleich zur vorherigen Lösung, welche sonst nur aufwändig vorbereitete Animationen zur Verfügung hatte.

Alle Anforderungen hinsichtlich Problemlösung und Weiterentwicklung werden in dieser Bachelorarbeit umgesetzt. Zudem werden die Performanceprobleme gelöst, sodass die Animation keine Ruckler mehr zeigt. Allerdings besteht immer noch Verbesserungspotential im Gebiet der Benutzerfreundlichkeit. Die Grafische Oberfläche der Orbitarium-Suite soll angepasst werden, sodass sie besser verständlich ist für erstmalige Benutzer. Zudem müssen die Projektoren ersetzt werden, da sie in nicht vollständig verdunkelten Räumen zu schwach sind. Diese Tätigkeiten können in Form einer weiterführenden Bachelorarbeit umgesetzt werden.

Abstract

In 2020, as part of a bachelor thesis, a software suite was created to run an Orbitarium. This software suite is a collection of different programmes that are needed to run the Orbitarium. Following it will be called Orbitarium-Suite. The Orbitarium itself is a plexiglass sphere onto which the surface of the Earth is projected to show various animations. It formerly belonged to the Technorama Winterthur and was also taken over by the ZHAW in 2020. The Orbitarium is currently located in a temporary room. In the future it will be exhibited on the ZHAW campus.

The version of the Orbitarium-Suite from the preliminary work has a performance problem. Played animations are displayed with approximately 3 frames per second and are therefore very stuttery. By making some adjustments in the code and by replacing a program of the suite, the performance is increased to about 15 frames per second. A climate simulation is implemented, which is based on a master thesis from 2010 and was later implemented in a Java application by Mr. Rege. In this bachelor thesis, the Java application is migrated to the Orbitarium-Suite to play it on a real globe instead of a virtual one. The Orbitarium-Suite is upgraded with the features to display animations from videos and web pages. These new capabilities make it easier to display new content compared to the previous solution, where only time-consuming pre-prepared animations were available.

All requirements regarding problem solving and further development are implemented in this bachelor thesis. In addition, the performance problems are solved so that the animation no longer shows stutters. However, there is still room for improvement regarding user-friendliness. The graphical user interface of the Orbitarium-Suite should be changed so that it is easier to understand for first-time users. In addition, the projectors need to be replaced as they are too weak in rooms that are not completely darkened. These activities can be implemented during an advanced bachelor thesis.

Vorwort

Das Orbitarium und die verschiedenen Darstellungen auf der Oberfläche sind für uns äusserst interessant. Es ermöglicht uns Animationen aus nächster Nähe zu betrachten. Die Themen Visual Computing und Unity interessieren uns zudem beide. Erfahrungen konnten wir bereits aus dem privaten Umfeld mitbringen, aber auch während dem Studium sammeln.

Ohne Unterstützung wäre diese Bachelorarbeit nicht möglich gewesen. Wir möchten uns ganz herzlich bei folgenden Personen bedanken:

- **Bei Herrn Prof. Dr. Karl Rege**: Für diese spannende Bachelorarbeit und die Unterstützung bei der Durchführung.
- **Bei Herrn Remo Preus und Nicolas Schaller**: Für die aufgewendete Zeit der Einführung in die Welt des Orbitariums und natürlich für ihre Bachelorarbeit, welche als Basis unserer dient.
- **Bei Herrn Robert Brem**: Für den Zeitaufwand für die Erklärung der Klimasimulation seiner Masterarbeit und für die Bereitstellung der entsprechenden Unterlangen.

Inhaltsverzeichnis

1 Einleitung

Die ZHAW hat das Orbitarium vom Technorama Winterthur übernommen. Im Jahr 2020 wurde im Rahmen einer Bachelorarbeit eine Software-Suite erarbeitet, um das Orbitarium zu betreiben. [1] Ein Orbitarium besteht hauptsächlich aus einer grossen Plexiglaskugel und Projektoren. Die Projektoren projizieren ein Bild auf diese Kugel. Die Projektion muss vorher durch Software aufbereitet werden. Auch wurde eine Animation kreiert, welche die globale Covid19 Verbreitung im Zeitraffer auf dem Orbitarium visualisiert. Die verschiedenen Programme für die Animation und die Darstellung werden folgend als Orbitarium-Suite bezeichnet.

1.1 Ausgangslage

Die Orbitarium-Suite funktioniert, jedoch ist sie noch nicht fertig. Es gibt noch keine Möglichkeit Visualisierungen aus universellen Quellen darzustellen. Ebenfalls stimmt die Performance der Animationen noch nicht. Das bedeutet, die Animation ruckelt während des Abspielens. Die Daten der Covid19-Fallzahlen sind veraltet und es ist nicht beschrieben, wie diese zu aktualisieren sind.

1.2 Aufgabenstellung

In erster Linie sollen alle Programme der Orbitarium-Suite auf den aktuellen Stand gebracht werden.

Die bisherige Version der Orbitarium-Suite liefert während dem Abspielen von Animationen zu wenig Bilder pro Sekunde. Dadurch ruckelt die Animation. Daher soll das Problem analysiert und behoben werden.

Es soll möglich sein, Visualisierungen aus universalen Quellen ohne grossen Aufwand auf den Globus projizieren zu können. Animation sollen Beispielsweise aus Video und Websites einfach auf dem Orbitarium dargestellt werden.

Es existiert eine Animation, welche die Auswirkung der Treibhausgaskonzentration auf das Klima aufzeigt. [2] Sie basiert auf einer Masterarbeit von Robert Brem und wurde von Herr Karl Rege erstellt. [3] Diese Animation soll in der Orbitarium-Suite implementiert werden, um sie auf dem Globus darzustellen.

2 Konzept

Der erste Teil der Arbeit besteht aus Einarbeitung in das bestehende Projekt. Zum einen gilt es zu verstehen wie das Orbitarium funktioniert, zum anderen müssen die Softwarekomponenten zwecks Entwicklung auf den Computern der Studenten eingerichtet werden. Dabei sind einige Sonderheiten zu beachten, die durch die Erfahrung in dieser Arbeit festgestellt wurden.

2.1 Aufbau Orbitarium Hardware[1]

Der Aufbau der Orbitarium Hardware ist nicht Teil dieser Arbeit und wird deshalb aus der vorhergehenden Arbeit [1], welche dies detailliert erklärt, in diesem Kapitel zitiert.

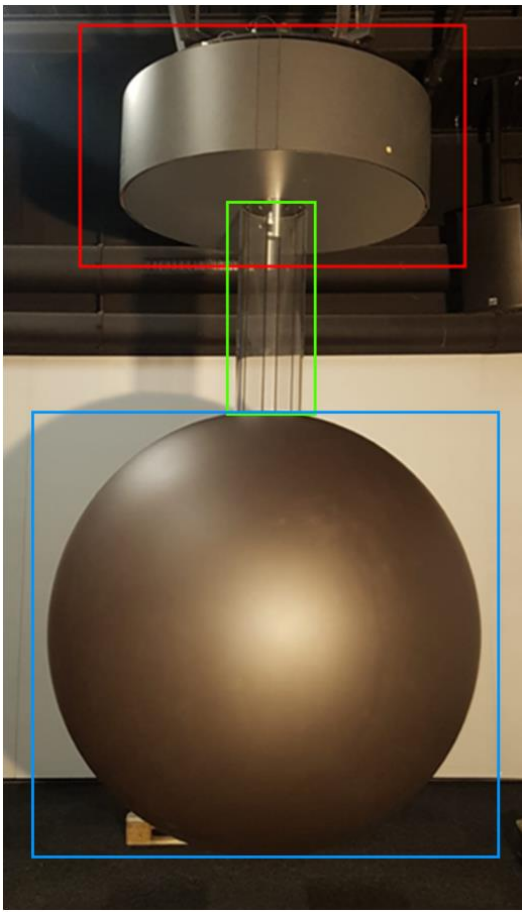


Abbildung 1: Orbitarium

Das Orbitarium ist eine an der Decke aufgehängte Konstruktion. Es ist in Abbildung 1 dargestellt.

Es besteht im Wesentlichen aus den 3 folgenden Teilen:

- Das Projektorgehäuse (zuoberst, rot umrandet)
- Die Plexiglasröhre (mittig, grün umrandet)
- Die Projektionskugel (unten, blau umrandet)

[1] Das Kapitel "2.1 Aufbau Orbitarium Hardware" ist komplett von der vorherigen Bachelorarbeit [1] zitiert

2.1.1 Das Projektorgehäuse

Das Projektorgehäuse enthält hauptsächlich die Beamer, welche «gegeneinander» projizieren und so ihr Bild auf die Projektionskugel werfen. Die genaue Projektionsgeometrie, also die Art und Weise, wie das Bild der Projektoren auf die Projektionskugel gelenkt wird, ist Teil der Analyse und wird im Kapitel 3.1.1 der Arbeit Neuartiges Orbitarium Darstellungskonzept [1] erklärt.

2.1.2 Die Plexiglasröhre

Die Plexiglasröhre verbindet das Projektorgehäuse mit der Projektionskugel. In dunklen Umgebungen ist sie praktisch nicht sichtbar, weswegen beim Betrachter das Bild eines in der Luft schwebenden Globus entsteht. Um das Gewicht der Projektionskugel tragen zu können, ist die Plexiglasröhre mit Stangen aus schwarz lackiertem Stahl verstärkt.

2.1.3 Die Projektionskugel

Die Projektionskugel ist eine Kugel aus mattem, schwarzem Plexiglas. Im Boden sitzt ein Spiegel, welcher das Licht aus der Plexiglasröhre reflektiert und auf die Projektionskugel umlenkt. Die Kugel ist schwarz lackiert, um beim Betrachten nicht zu viel direktes Licht vom Spiegel durchscheinen zu lassen.

2.2 Aufbau Orbitarium-Suite

In den folgenden Unterkapiteln werden die Bestandteile der Orbitarium-Suite erklärt. Die Orbitarium-Suite bezeichnet alle Programme, welche dafür verantwortlich sind, Inhalte auf dem Orbitarium darzustellen. Die Sammlung beinhaltet sowohl selbst programmierte wie auch fremde Software. Sie wird in erster Linie von Mitarbeitern der ZHAW bedient. Fremde Personen interagieren höchstens über extra dafür bereitgestelltes Equipment mit dem Orbitarium, aber nie direkt mir der Suite.

2.2.1 Orbitarium Control Panel (Unity)

Das Orbitarium Control Panel ist die Steuerungszentrale für das Orbitarium. Hier werden Animationen für das Orbitarium gestartet und gesteuert. Es gibt bisher nur eine vorbereitete Animation. Diese stellt die globale Covid19 Verbreitung dar.

2.2.2 Animation Service

Der Animation Service bietet die Möglichkeit eigene Animationen zu kreieren. Es handelt sich um eine in C# geschriebene Webapplikation. Diese bietet ein REST-Interface mit verschiedenen Funktionen an. Grundsätzlich geht es darum Daten aus einer Quelle zu importieren und anschliessend aus den importierten und aufbereiteten Daten eine Animation zu generieren. Dadurch können Animationen sehr leicht mit aktuelleren Daten versorgt werden.

2.2.3 Animations

In diesem Verzeichnis werden die vom Animation Service vorbereiteten Animationen gespeichert. Animations selbst wird ebenfalls als Webservice betrieben. Das Orbitarium Control Panel greift via HTTP auf die Animationen zu.

2.2.4 Spacedesk

Spacedesk stellt einen virtuellen Monitor zur Verfügung. Die Auflösung kann dabei beliebig eingestellt werden. Für das Orbitarium wird ein quadratischer Monitor mit der Auflösung 2160 x 2160 benötigt.

2.2.5 Browser / Browserengine

Animationen, welche mit dem Webservice erstellt werden, werden im Browser abgespielt. Das Control Panel startet dafür via Chromedriver den Google Chrome mit dem entsprechenden URL, verschiebt das Browserfenster auf den Spacedesk Monitor und aktiviert den Vollbildmodus. Die Animation aus dem Browser wird mithilfe einer Virtualcam ins Controlpanel übertragen und anschliessend auf dem Globus dargestellt.

2.2.6 Virtualcam

Mithilfe von OBS-Studio wird der virtuelle Monitor von Spacedesk bzw. das Browserfenster darin aufgenommen. Die Aufnahme kann entweder als Video gespeichert oder an eine virtuelle Webcam (Virtualcam) gestreamt werden. Diese Virtualcam wird vom Orbitarium Control Panel erfasst und als Input verwendet.

2.2.7 Entwicklungsumgebungen

Der Animation Service und das Orbitarium Control Panel werden im Visual Studio 2019 entwickelt. Für die UI Elemente im Control Panel wird Unity verwendet.

2.3 Spezielle Aspekte bei der Inbetriebnahme

Hier werden spezielle Eigenheiten beschrieben, die während der Inbetriebnahme der Orbitarium-Suite auf einem neuen Computer berücksichtigt werden müssen. Folgende Software muss heruntergeladen und installiert werden:

- Unity Hub
- Unity (Version 20.3 LTS)
- Visual Studio 2019
- OBS-Studio
- OBS Virtualcam Plugin
- Google Chrome
- Chromedriver
- IIS
- Powershell 7
- Python 3

2.3.1 Repositories

Im Anhang sind drei Git Repositories zu finden. Alle drei Repositories müssen entweder mit Git auf den Computer geklont oder als ZIP-Datei heruntergeladen und entpackt werden.

2.3.2 Unity

Zuerst muss Unity Hub heruntergeladen und installiert werden. Als nächstes wird das von GitHub heruntergeladene Verzeichnis «Orbitarium-Control-Panel» im Unity Hub hinzufügefügt. Die benötigte Unity Version, aktuell 20.3 LTS, kann anschliessend installiert werden. Visual Studio wird im Installer vorgeschlagen und kann gleich mitinstalliert werden. Alternativ kann Visual Studio auch separat installiert werden.

Das Orbitarium-Control-Panel muss nun im Unity Editor gestartet und im Ordner Orbitarium-Control-Panel-Build ein Build erstellt werden. Ein Build ist dabei eine plattformspezifisch kompilierte Version des Programms, die von Unity erstellt wird, damit es ohne den Unity Editor läuft.

2.3.3 Google Chrome

Google Chrome muss heruntergeladen und installiert werden, da verschiedene Animationen einen Browser benötigen. Es könnten auch andere Browser verwendet werden, jedoch müsste der Code des Control Panels dann entsprechend angepasst werden.

2.3.4 Chromedriver

Damit das Orbitarium Control Panel mit dem Chrome interagiert werden kann, muss der Chromedriver heruntergeladen werden. Wichtig dabei ist, die Version des Chromedrivers muss zur Version des Chromes passen. Die neuste Chromedriver Version passt in der Regel mit dem neusten Chrome zusammen.

Damit der Chromedriver vom Orbitarium Control Panel gefunden wird, muss dieser zum einen in den Assets im Unity-Editor zum anderen im Verzeichnis «Orbitarium-Contol-Panel-Build/Orbitarium Control Panel_Data/Managed» abgelegt werden.

2.3.5 Animation Service vorbereiten

Zunächst wird das Visual Studio Projekt im von GitHub heruntergeladenen Verzeichnis Animation-Service geöffnet. In den Projekteinstellungen muss der Pfad zum Projektverzeichnis angepasst werden. Konkret bedeutet das, in der Datei «Web.config» bedarf es einer Anpassung des Wertes der Variable «BASE_DIR».

Anschliessend wird das Projekt im Releasemodus gebuildet und danach gepublisht. Das Builden im Releasemodus ist wichtig, da das Publishen die Releasebuilds anstelle der Debugbuilds verwendet.

2.3.6 IIS

Der Animation Service wird auf dem IIS Webserver ausgeführt. Dieser kann in den Windows Einstellungen aktiviert werden. Dabei muss das .NET 4.2 Feature ebenfalls aktiviert werden.

Um den Animation Service im IIS einzurichten, wird im IIS eine neue Webapplikation mit dem Pfad zur publishten Applikation hinzugefügt. Dieser benötigt Port 12345.

Das von GitHub heruntergeladene Verzeichnis «Orbitarium Animations/animations» muss ebenfalls im IIS als neue Webapplikation hinzugefügt werden. Hier wird Port 12346 verwendet.

2.3.7 Corona Animation erstellen

Die folgenden Schritte erklären die erstmalige Inbetriebnahme der Corona Animation. Zuerst werden Daten zu Covid19, Ländergrenzen und Einwohnerzahlen in den Webservice importiert. Anschliessend wird die Animation im Webservice generiert. Nun kann die Animation verwendet werden.

2.3.7.1 CSV Updateskript

Die Covid19-Fallzahlen stehen für die erste Inbetriebnahme bereits aufbereitet als CSV-Dateien zur Verfügung. Die vorhandenen Daten sind zwar veraltetet, genügen jedoch um sich zu vergewissern, dass die Animation läuft. Für die Aktualisierung der Covid19-Fallzahlen existiert ein Skript, welches in Kapitel 3.1.2 genauer erklärt ist. Damit das Skript funktioniert müssen darin die Pfade zum Animation Service und Animations angepasst werden. Ausserdem muss Powershell 7 aus dem Microsoft Store und Python 3 installiert werden.

2.3.7.2 Daten importieren

Die aufbereiteten CSVs mit den Covid19-Fallzahlen werden im Ordner «Orbitarium-Animation-Service/Animation_Service/WebApplication1/Data/source/corona_data» abgelegt. Mit Powershell wird nun der Animation Service aufgerufen. Dabei werden zuerst Einwohnerzahl und Ländergrenzen importiert. Die Powershell Befehle dazu sind in Code-Ausschnitt 1 und Code-Ausschnitt 2 ersichtlich. Die Daten dazu befinden sich bereits im «Data/Source» Verzeichnis des Animation Services.

```
Invoke-WebRequest -Uri http://localhost:12345/api/import -Method POST
-Body ("country_borders"|ConvertTo-Json) -ContentType "application/json"
```

Code-Ausschnitt 1: Ländergrenzen importieren

```
Invoke-WebRequest -Uri http://localhost:12345/api/import -Method POST
-Body ("country_population"|ConvertTo-Json) -ContentType "application/json"
```

Code-Ausschnitt 2: Bevölkerungszahlen importieren

Im Anschluss können die Covid19-Fallzahlen importiert werden:

```
Invoke-WebRequest -Uri http://localhost:12345/api/import -Method POST
-Body ("corona_data"|ConvertTo-Json) -ContentType "application/json"
```

Code-Ausschnitt 3: Covid19-Fallzahlen importieren

2.3.7.3 Animation generieren

Mit den importierten Daten wird schliesslich die Animation generiert. Dafür wird der Animation Service wie in Code-Ausschnitt 4 ersichtlich aufgerufen.

```
Invoke-WebRequest -Uri http://localhost:12345/api/animation -Method POST
-Body ("generate,corona_spread"|ConvertTo-Json)
-ContentType "application/json"
```

Code-Ausschnitt 4: Animation generieren

2.3.8 OBS Studio

OBS Studio muss heruntergeladen und installiert werden. OBS Studio hat bereits eine virtuelle Webcam dabei. Diese wird aber von Unity nicht erkannt. Die Virtualcam muss als Plugin nachinstalliert werden. Dies reicht jedoch nicht. Damit diese von Unity erkannt wird müssen in der Registry zwei Einträge erstellt werden.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{860BB310-5D01-11d0-BD3B-
00A0C911CE86}\Instance\{27B05C2D-93DC-474A-A5DA-9BBA34CB2A9C}]
"DevicePath"="obs:virtualcam"

[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Classes\CLSID\{860BB310-5D01-11d0-
BD3B-00A0C911CE86}\Instance\{27B05C2D-93DC-474A-A5DA-9BBA34CB2A9C}]
"DevicePath"="obs-virtualcam"
```

Code-Ausschnitt 5: Dummy Einträge für Virtualcam

Die Virtualcam des Plugins kann anschliessend über die Menüleiste im OSB Studio gestartet werden. Unity erkennt diese dann automatisch.

3 Realisation

In diesem Kapitel sind sämtliche Änderungen an der Orbitarium-Suite dokumentiert.

3.1 Daten für Corona Visualisierung aktualisieren

Hier wird beschrieben, wie die Covid19 Daten zu aktualisieren sind.

3.1.1 Datenherkunft

Die Daten für die Corona Visualisierung stammen von www.worldmeters.info. Um die Daten der Website zusammenzutragen wird ein Jupyter Notebook verwendet. Dieses Notebook wurde extra für diesen Zweck erstellt und stammt von einem Github Repository. [4]

3.1.2 Automatisierung

Jupyter Notebooks sind für manuelle Ausführung gedacht. Um eine automatisierte Ausführung zu ermöglichen, lässt sich das Jupyter Notebook mit dem folgenden Befehl in ein Python Skript konvertieren. Daraus resultiert das Pythonskript «clean_data.py»

```
jupyter nbconvert --to script clean_data.ipynb
```

Code-Ausschnitt 6: Jupyter Notebook in ein Pythonskript konvertieren

Um den ganzen Prozess des Corona-Daten-Updates zu automatisieren, wurde ein PowerShell Skript geschrieben. Es löscht zu Beginn alle CSV-files des letzten Durchgangs. Als nächstes wird das Python Skript ausgeführt. Das Ergebnis ist eine Sammlung mehrerer CSV-Files. Eines dieser Files, nämlich «covid19_confirmed_usa.csv» muss zusätzlich nach UID sortiert werden. Dafür wird es mit dem PowerShell Befehl aus Code-Ausschnitt 7 eingelesen, sortiert und wieder ausgegeben. Damit beim Sortieren die Reihenfolge stimmt, muss die Spalte UID als Ganzzahl interpretiert werden und nicht als String. Ansonsten werden die Daten falsch sortiert. Zusätzlich wird das Trennzeichen von Komma zu Semikolon geändert. Das behebt das Problem, dass gewisse Werte ein Komma enthalten und dadurch vom Animation Service falsch behandelt werden.

```
Import-Csv .\temp.csv | Sort-Object {[int]$_.UID} | Export-Csv
-Path .\time_series_covid19_confirmed_US.csv -Delimiter ';' -UseQuotes Never
```

Code-Ausschnitt 7: CSV Datei sortieren

Nachdem die CSV-Files aufbereitet wurden, werden sie in die richtigen Verzeichnisse kopiert. Wie in Code-Ausschnitt 8 ersichtlich ist, unterscheiden sich die Pfade von Computer zu Computer und müssen daher angepasst werden.

```
AnimationServiceFolder = "C:\Users\Tim\BA\Orbitarium-Animation-
Service\Animation_Service\WebApplication1\Data\source\corona_data"
$AnimationFolder = "C:\Users\Tim\BA\Orbitarium-Animations\animation\co-
rona_spread\data"

Copy-Item full_grouped.csv -Destination $AnimationFolder

Copy-Item full_grouped.csv -Destination $AnimationServiceFolder
Copy-Item time_series_covid19_confirmed_US.csv
-Destination $AnimationServiceFolder
Copy-Item covid_19_clean_complete.csv -Destination $AnimationServiceFolder
```

Code-Ausschnitt 8: CSV Dateien kopieren

Mit dem Code-Ausschnitt 9 werden die Coronadaten aus den CSV-Files importiert und die Animation generiert.

```
Invoke-WebRequest -Uri http://localhost:12345/api/import -Method POST
-Body ("corona_data"|ConvertTo-Json) -ContentType "application/json"

Invoke-WebRequest -Uri http://localhost:12345/api/animation -Method POST
-Body ("generate,corona_spread"|ConvertTo-Json)
-ContentType "application/json"
```

Code-Ausschnitt 9: Daten importieren und Animation generieren

Das Powershell Skript muss mit Powershell 7 ausgeführt werden. Grund dafür ist: Der Parameter «-Delimitter» des Befehls «Export-Csv» aus Code-Ausschnitt 7 existiert in der standardmässig installierten Powershell Version 5 nicht. Installiert wird es aus dem Microsoft Store.

3.2 Videoquelle

Um nicht nur mit der Virtualcam Input aufzunehmen, wurde die Anforderung gestellt Video Input in der Applikation darzustellen. Im folgenden Kapitel wird die Implementation dazu besprochen.

3.2.1 Analyse

Die Möglichkeit, Videos mit der Virtualcam aufzunehmen und so auf dem Globus darzustellen besteht bereits. Dies setzt aber voraus, dass das Video auf den Virtuellen Desktop gezogen wird und dann korrekt ausgerichtet wird. Ein Video auf diesem Weg abzuspielen ist nicht nur sehr aufwendig, sondern auch für den Benutzer nicht offensichtlich. Es soll möglich sein, direkt vom Orbitarium Control Panel ein Video auszuwählen und abzuspielen.

Im Orbitarium Control Panel wird die Darstellung der Erdoberfläche über eine Textur gelöst. Eine Textur ist dabei nur ein Bild, welches in Unity weiterverarbeitet werden kann. [5] Diese Textur wird dann mit einem Shader bearbeitet, um sie auf dem Orbitarium darzustellen. Ein Shader ist dabei ein Grafikprogramm, welches die Pixelfarbe auf dem Bildschirm bestimmt. [6]

Unity bietet ein Feature an, welches es erlaubt ein Video darzustellen. Dazu kann eine Videodatei eingelesen werden und als «RenderTexture» angezeigt werden. Eine «RenderTexture» ist eine Textur, welche dynamische Inhalte anzeigen kann. [7] So kann zum Beispiel ein Videostream dieser Textur weitergegeben und dann weiterverarbeitet werden.

3.2.2 Implementation

Um das Video der «RenderTexture» zu übergeben, wird ein GUI benötigt, um eine Datei auszuwählen. Zuerst wurde versucht diese Funktion mit den nativen Unity Klassen zu implementieren. Es hat sich aber herausgestellt, dass diese Funktionen nicht im Buildmodus funktionieren. Deshalb musste eine zusätzliche GUI Komponente her.

Da der Aufwand zur Entwicklung einer solchen Komponente zu gross ist, wurde eine aus dem Unity Asset Store genommen. Der Unity Asset Store ist eine Ansammlung an kostenlosen und kostenpflichtigen Assets, welche für das eigene Projekt eingesetzt werden können. [8] Um einen Dialog anzuzeigen, wurde das «Simple File Browser» Asset verwendet. Mit diesem Asset kann zur Laufzeit ein Dialog angezeigt werden, damit der Benutzer eine Datei auswählen kann. [9] Nach Auswahl der Datei wird der Pfad dann an die «RenderTexture» übergeben und das Video wird dargestellt. In Abbildung 2 wird der «Simple File Browser» zur Laufzeit gezeigt.

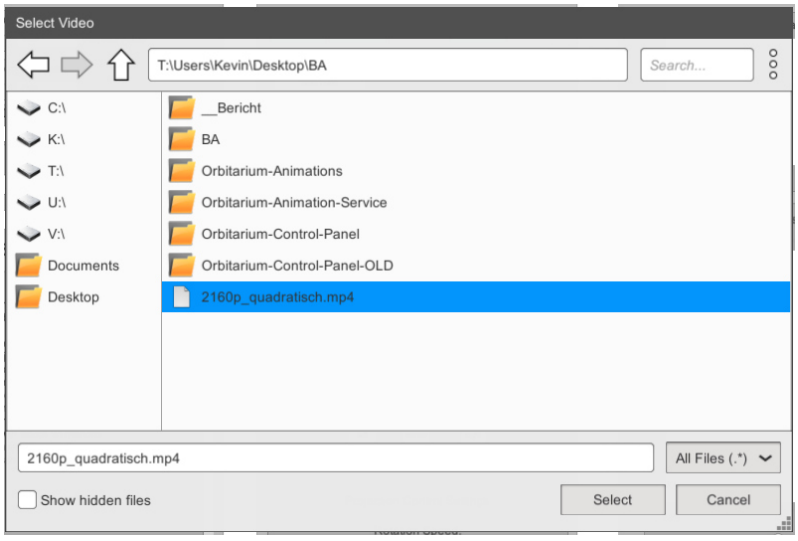


Abbildung 2 Simple File Browser

Um das Video korrekt starten und stoppen zu können, müssen die entsprechenden Buttons im Orbitarium Control Panel implementiert werden. Dabei wurde das Unity UI Event System verwendet. Dadurch wird die «RenderTexture» über Buttons im GUI gesteuert. Abbildung 3 zeigt die entsprechenden Buttons in der Applikation.

Abbildung 3 GUI für Video Controls

3.3 Performance

Die Performance des Orbitarium Control Panel war zum Zeitpunkt der letzten Entwicklung der vorherigen Bachelorarbeit nicht optimal. Dies hat sich vor allem beim Rotieren der Animation gezeigt. Dort ist die zu geringen Anzahl Frames pro Sekunde am deutlichsten sichtbar. Dies hat sich in Form störender Ruckler in der Animation gezeigt. In den folgenden Kapiteln wird die Analyse der Performance und die Implementation der neuen Lösung beschrieben.

3.3.1 Analyse der Performance

Unity bietet die Möglichkeit Statistiken über die CPU-Auslastung, GPU-Auslastung und Frames per Second (FPS) darzustellen. Diese Statistiken können eingeblendet werden, wenn Unity im Editor gestartet wird. [10]

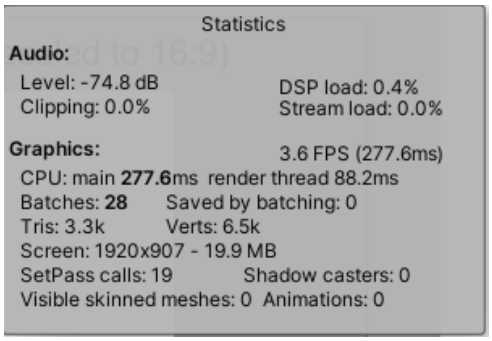


Abbildung 4 Unity Statistiken der alten Implementation

In Abbildung 4 sind die Statistiken der alten Implementation zu erkennen. Diese Statistiken basieren auf der Ausführung des Orbitarium Control Panel im Editor auf einem Computer der Studenten mit Mittelklasse Hardware. Diese Hardware ist wesentlich langsamer als die Hardware des Orbitarium Computers in Winterthur. Das Ziel ist es, auf der Hardware der Studenten eine vernünftige Performance zu erzielen. Somit wird sichergestellt, dass die Performance auf dem Orbitarium Computer ausreichend ist.

Die Statistiken in Abbildung 4 zeigen klar, dass die Performance ungenügend ist. Die Applikation läuft mit 3.6 FPS und die CPU benötigt durchschnittlich 278 Millisekunden, um ein Frame zu berechnen. Diese Zeit beinhaltet den Update Loop aller Unity Objekte, um den aktuellen Frame anzuzeigen. [11] Anhand dieser Statistik kann noch keine genügend gute Aussage gemacht werden, wo die Ursache des Problems liegt. Unity bietet hierfür aber ein gutes Tool an, den Unity Profiler.

Der Unity Profiler ist ein Tool, um Performance Informationen über die Applikation zu erhalten. Der Profiler zeigt Informationen über die GPU- und CPU-Auslastung, den verwendeten Speicher, den Renderer und das Audio an. [12] Ausserdem werden auch Performance Informationen zu den einzelnen Code Stellen angezeigt. Dieses Feature ist in diesem Fall sehr nützlich, um festzustellen, wo sich der Performanceverlust befindet.

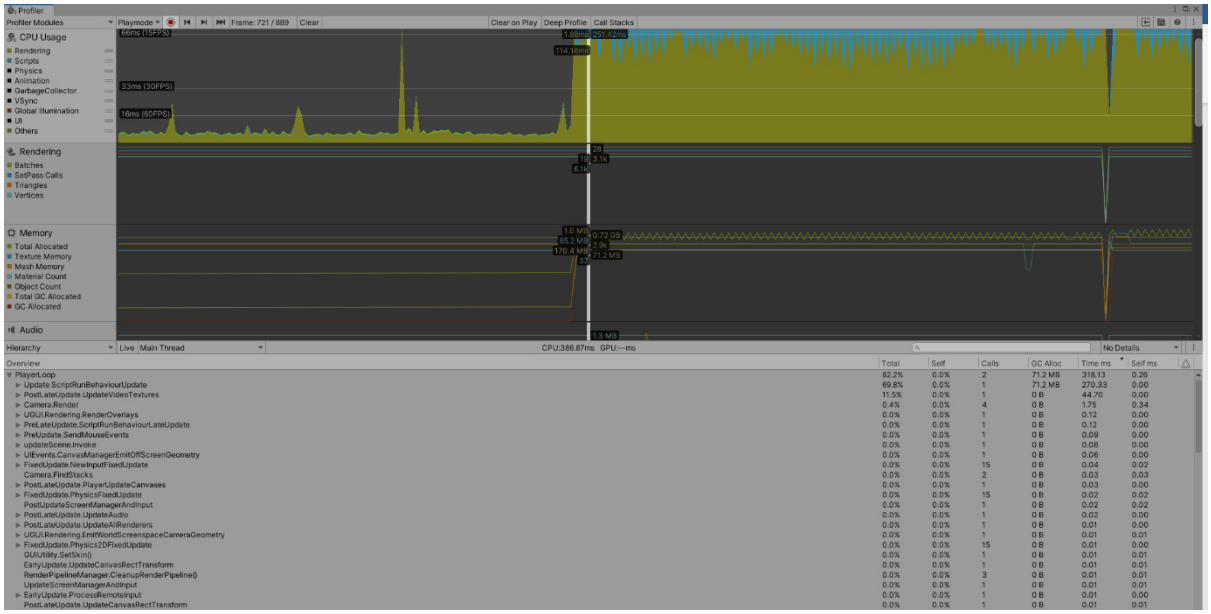


Abbildung 5 Profiler Übersicht der Applikation

In Abbildung 5 ist das Fenster des Profilers zu sehen. In der oberen Hälfte befindet sich eine grafische Übersicht, über die Auslastung der einzelnen Komponenten basierend auf der Zeit, wie lange diese Komponenten beschäftigt sind. Im unteren Teil befindet sich eine Auflistung der Klassen und Funktionen zusammen mit der Zeit, die Unity braucht, um diese Funktionen auszuführen.

In Abbildung 5 lässt sich erkennen, dass nach einem bestimmen Zeitpunkt die Auslastung der CPU sehr hoch ist. Der Grund für die hohe CPU-Auslastung ist das Drücken des Buttons «Render Displays» im Orbitarium Control Panel. Dieser Button führt dazu, dass die Virtualcam in der Applikation dargestellt wird. In Abbildung 6 erkennt man zusätzlich, dass 70% der Zeit für die Ausführung des Codes in «RenderHandler.Update()» verwendet wird. Diese Methode ist insbesondere für das Rendern der Virtualcam in Unity verantwortlich. Daraus lässt sich schliessen, dass der Performanceverlust durch den Code in dieser Methode entsteht.

Abbildung 6 Unity Profiler Codeübersicht

In Code-Ausschnitt 10 befindet sich Code aus der Klasse «RenderHandler.cs». In diesem Code wird der Input der Virtualcam zugeschnitten. Aufgrund der vorherig erklärten Performanceanalysen lässt sich erkennen, dass der Performanceverlust in dieser Funktion entsteht.

Laut einer Aussage im Unity Forum [13] ist die Performance der «WebcamTexture.GetPixels()» Funktion sehr schlecht. Dies ist der Fall, weil in jedem Frame das Pixel Array erneut angelegt werden muss und der GPU übergeben wird. Dazu kommt noch der Overhead der Webcam Klasse dazu, welcher die Performance noch zusätzlich verschlechtert. Da es sich bei unserem Webcam Output der Virtualcam um ein sehr grosses Bild handelt, ist diese Funktion nicht zu empfehlen.

```
void Update()
{
    {
        color = inputFeedTexture.GetPixels(pixelsToCut, 0, inputResY, inputResY);
        Texture.SetPixels(color);
        Texture.Apply();
        //inputImage.Texture = Texture;
        transformedImage.Texture = Texture;
    }
}
```

Code-Ausschnitt 10: Ausschnitt der Klasse RenderHandler

3.3.2 Performance Verbesserung

Eine alternative Implementierung, ist das Abschneiden des Bildes auf einen Shader zu verlagern. Ein Shader kann die Pixel eines Bildes sehr schnell bearbeiten und da die Animation sowieso mit einem Shader bearbeitet wird, um sie korrekt auf dem Globus darzustellen, entsteht nur ein kleiner Overhead, da die Shader kombiniert werden können.

```
float CutPixels(float x) {
// cut left and right border
    if (_IsSquare == 1)
    {
        float pixelsToCut = (_InputX - _InputY) / 2;
        float factor = _InputY / _InputX;

        return (pixelsToCut / _InputX) + x * (factor);
    }

    return x;
}
```

Code-Ausschnitt 11: Wegschneiden schwarzer Balken

In Code-Ausschnitt 11 ist die neue Implementation des Abschneidens der Pixel zu erkennen. Hier wurde zusätzlich noch eine Variable implementiert, um das Abschneiden zu umgehen. Dies ist von Vorteil, wenn der Input bereits quadratisch ist. Dann kann die ganze Wegschneideprozedur weggelassen werden, was eine weitere Leistungssteigerung mit sich bringt.

Der Code im Shader bearbeitet nur die X-Achse des Frames, da nur die Balken links und rechts weggeschnitten werden müssen. Im Shader bewegt sich die X-Variable im Bereich 0 bis 1. Das heisst, um die Balken korrekt wegzuschneiden muss von der originalen Auflösung, zusammen mit der Breite der Balken eine Umwandlung berechnet werden. Die Abbildung 7 visualisiert diese Umrechnung.

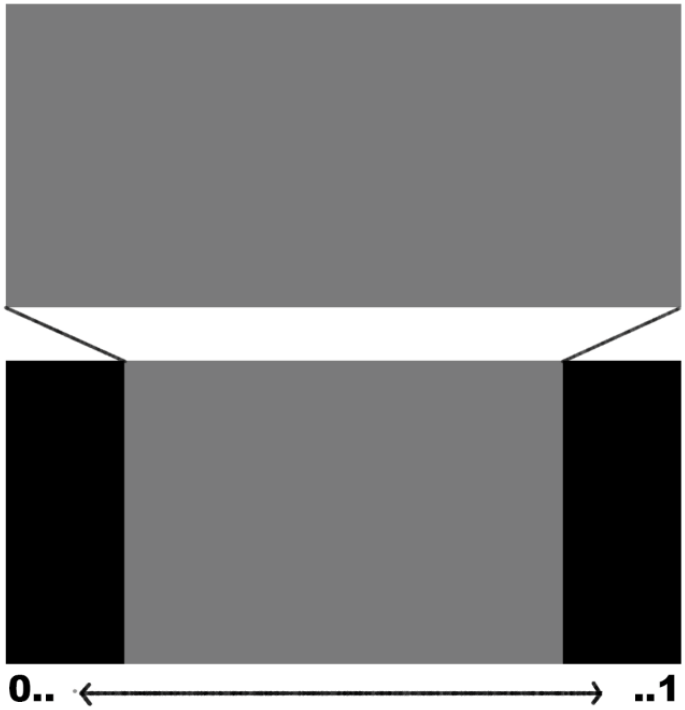


Abbildung 7 Visualisierung Shader Code

Der Punkt nach dem linken Balken muss auf der Ziel-Textur an der Stelle 0 liegen. Der Punkt vor dem rechten Balken soll dann auf der Stelle 1 liegen. Damit das Bild dann auf die Ziel-Textur passt, wird diese mit einem Faktor gestreckt. Dieser errechnet sich aus der Länge und Breite des Ursprungs Bildes.

3.3.3 Vergleich der Performance Verbesserung

Aus der erneuten Messung der Performance mit Unity ist ersichtlich, dass die FPS nun wesentlich höher sind, wie in Abbildung 8 zu sehen ist. Von der alten Implementation mit 3-4 FPS auf 13-15 FPS mit der neuen Implementation. Dies reicht schon, um auf Mittelklasse Hardware ein flüssiges Bild zu erzeugen.

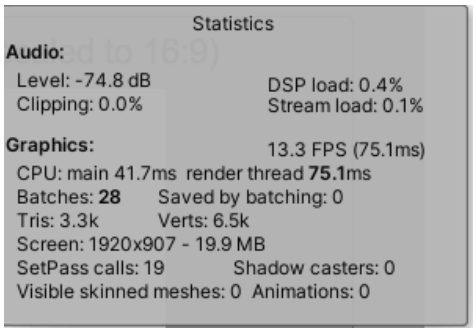


Abbildung 8 Performance nach der neuen Implementation

3.4 Virtualcam Software

Die ursprüngliche Orbitarium-Suite setzt als Virtualcam Software Manycam ein. Da die Auflösung und die Animation auf dem virtuellen Monitor quadratisch sind, ManyCam aber nur Output im 16:9 Format liefert, müssen die dadurch links und rechts entstandenen schwarzen Balken abgeschnitten werden. Aus diesem Grund wurde ManyCam durch OBS-Studio ersetzt, das beliebige Bildschirmformate unterstützt.

3.5 Überschneidung Projektoren

Im Orbitarium Control Panel wurde für die beiden Bilder, welche an die Projektoren gesendet werden, jeweils ein Teil des anderen Bildes miteinbezogen. Dies führt zu einer Überschneidung der Inhalte und äussert sich auf dem Globus, indem dieser Teil heller erscheint. Auf dem Globus wird die Trennlinie der beiden Projektoren dadurch sehr deutlich. Abbildung 9 zeigt die Überschneidung, wie sie in der Applikation sichtbar ist. Der Teil im roten Rechteck wird jeweils doppelt dargestellt.

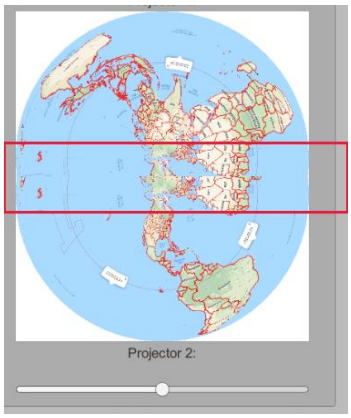


Abbildung 9 Überschneidung in der Applikation

Um die Überschneidung regulieren zu können, wird ein Regler implementiert. Dieser ist in Abbildung 9 unterhalb des Bildes sichtbar. Mit diesem Regler können die Bilder verschoben werden, bevor sie and die Projektoren gesendet werden. Dieser Regler muss aber mit der Steuerung der Projektoren zusammen eingestellt werden, da das Verschieben der Bilder auch das Verschieben und Neuausrichten der Projektoren benötigt. Die Einstellungen für den Regler, sowie die Einstellungen der Projektoren werden gespeichert, sodass man eine genügend gute Einstellung für den nächsten Neustart speichern kann. Abbildung 10 lässt erkennen, wie die Applikation die Verschiebung darstellt. In diesem Beispiel wurde die Überlappung komplett weggelassen.

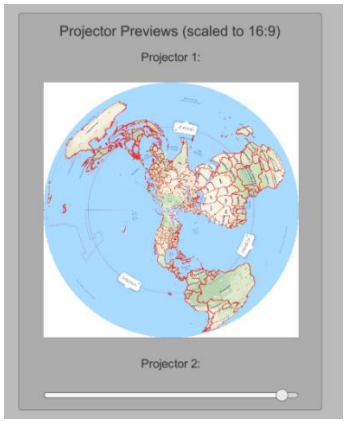


Abbildung 10 Angepasste Überschneidung

3.6 Echtzeit Simulation der Auswirkung der Treibhausgaskonzentration auf das Klima

In der Masterarbeit von Herrn Robert Brem [3] wurde ein Modell zur Simulation der Auswirkung der Treibhausgaskonzentration auf das Klima erstellt. Eine simple Version dieses Modells wurde in einer Applikation von Herrn Karl Rege zur Simulation in Echtzeit implementiert.

3.6.1 Klimamodell

Die Berechnung des Klimamodells ist in drei Bereiche unterteilt.

Der erste Bereich bezieht sich auf die Darstellung der Küstenlinien, wo befindet sich Wasser und wo Land. Die Meerfärbung ist hauptsächlich von der Tiefe des Meeres abhängig. Die Vegetation ist der zweite Bereich der Darstellung. Sie bestimmt wo sich dunkelgrüner Regenwald oder hellbraune Wüste befindet. Die Vegetation ist von der Temperatur in Bodennähe sowie vom Niederschlag abhängig. Der dritte und letzte Bereich der Darstellung ist die Eisbedeckung der Erde. Sie bestimmt wo sich Eis auf der Erde befindet. [3]

Die Berechnung der Temperatur ist ein wichtiger Bestandteil des Modells. Für die Darstellung eines solchen komplexen Systems wurde ein Casual Loop Diagramm (CLD) verwendet. Dieses stellt Zusammenhänge zwischen beeinflussenden Variablen grafisch dar und eignet sich hervorragend für das Temperatursystem. [3] In Abbildung 11 ist das CLD der Temperatur zu erkennen, welches für das Modell verwendet wird.

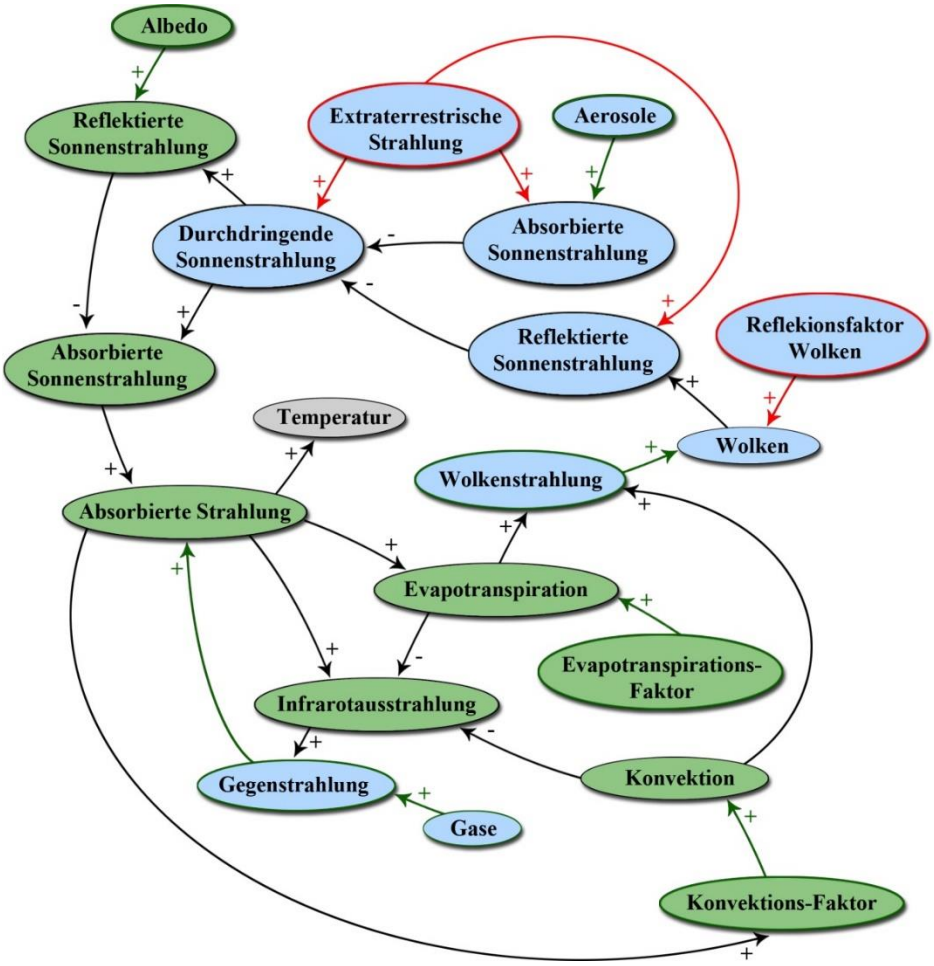


Abbildung 11 CLD der Temperatur [3]

3.6.2 ClimaSimul Applikation

Zur Zeit der Entwicklung der Masterarbeit war das Modell zu komplex, um es in Echtzeit darzustellen. Von Herrn Karl Rege und Adrian Meyer wurde eine simple Version des Modells zur Visualisierung umgesetzt, um es dennoch vollständig in Echtzeit darzustellen. Das Modell zeigt die Erde in der Gegenwart bis zum Jahr 10000 in der Zukunft. [2] Das Programm zeigt den CO2 Gehalt, welcher eingestellt werden kann, die Temperatur und die Veränderung der Höhe des Meeresspiegels.

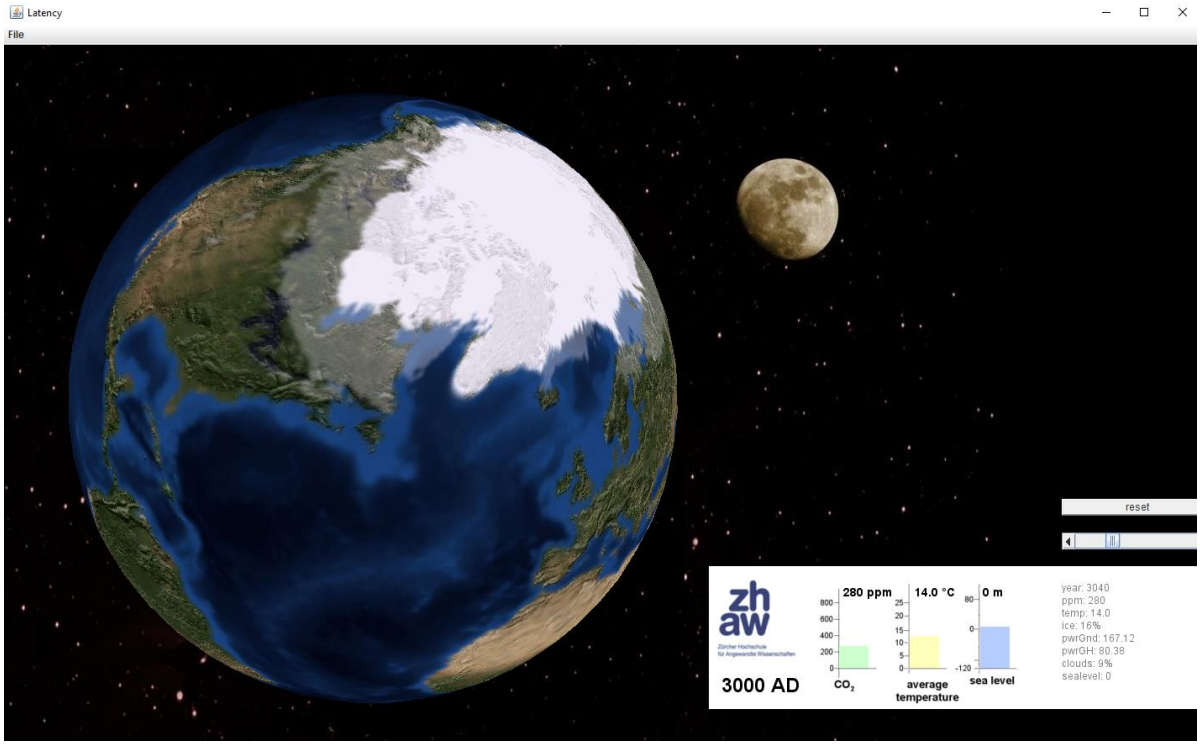


Abbildung 12 ClimaSimul Programm

3.6.3 Implementation in Unity

Um die Co2 Animation auf dem Orbitarium darzustellen, muss eine neue Lösung erarbeitet werden. Die Möglichkeit, das aktuelle Java Programm mit der Webcam aufzuzeichnen wurde schnell verworfen, da diese Möglichkeit zu umständlich für den Benutzer ist. Um die Co2 Animation einfacher auf dem Orbitarium darzustellen, soll diese in Unity implementiert werden.

Durch die Ähnlichkeit von Java und C#, kann der Code der Klimasimulation einfach in C# Scripts umgewandelt werden. Abbildung 13 stellt das konzeptuelle Design der Co2 Implementierung dar. Die Klasse Earth berechnet die Temperatur anhand der Treibhausenergie und der Reflektierten Sonnenstrahlen. Diese Energie wird anhand der Gletscher berechnet, welche aus der Klasse «Glaciers» kommen. Die Klasse «Glaciers» berechnet wieviel Prozent des Planeten mit Gletschern überdeckt sind und wie viel Energie dadurch reflektiert wird.

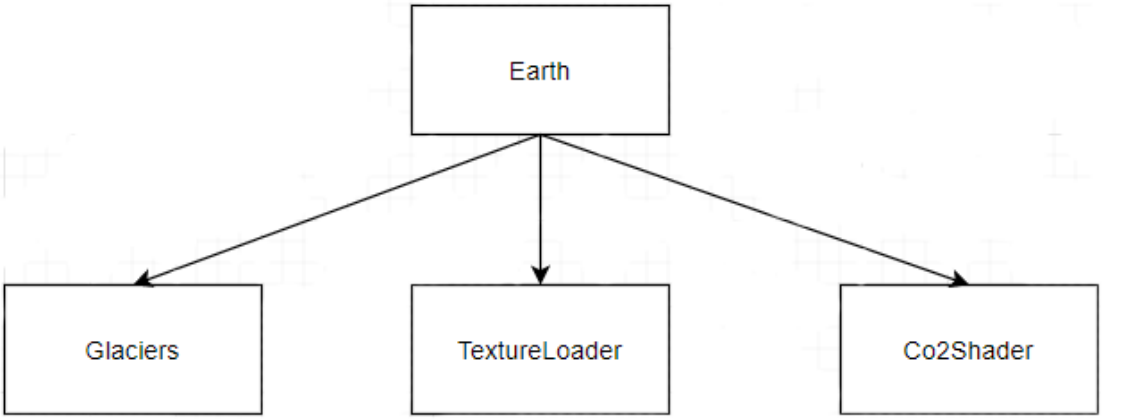


Abbildung 13 Konzeptuelles Design der Implementierung

Nachdem die Temperatur in der Klasse Earth berechnet wurde, wird die Temperatur an die «TextureLoader» Klasse weitergegeben. Die «TextureLoader» Klasse gibt das korrekte Bild zurück, aus der Liste der in der Masterarbeit berechneten Bilder. In Unity werden die Bilder zwecks Performanz zuerst geladen, damit sie später schneller ausgetauscht werden können. Nach dem Laden der Bilder werden die neusten zwei Bilder jeweils and den Co2Shader übergeben. Dieser Shader überblendet dann das neuste Bild, um es im Orbitarium Control Panel anzuzeigen.

3.6.4 Überblendung der Bilder

Das Überblenden der Bilder ist nötig, da es keine Animation gibt, sondern nur eine begrenzte Anzahl Bilder. Daher müssen die beiden passenden Bildern überblendet werden, um eine flüssige Animation zu erhalten. In Unity wurde die Überblendung mithilfe eines Shaders umgesetzt. Code-Ausschnitt 12 zeigt den Shader Code des Fragment Shaders. Anhand zweier Texturen und einem Alpha Wert, welche von der Klasse Earth übergeben werden, wird die Überblendung berechnet.

```
fixed4 frag(v2f i) : SV_Target
{
    fixed4 col = tex2D(_MainTex, i.uv);
    fixed4 col2 = tex2D(_Texture2, i.uv);
    col *= 1 - _Alpha;
    col2 *= _Alpha;

    return col + col2;
}
```

Code-Ausschnitt 12: Shader Code des Co2 Shader

Code-Ausschnitt 13 lässt erkennen, wie die Werte für den Shader gesetzt werden. Durch «Graphics.Blit()» wird mithilfe des Shaders eine neue Textur erstellt. «Graphics.Blit()» kopiert eine Textur in eine neue und wendet dabei einen Shader an. [14] Danach wird die erstellte Textur an einen weiteren Shader weitergegeben, der verantwortlich ist das Bild auf dem Orbitarium darzustellen.

```
TextureLoader.ComputeTexturs();

mat.SetTexture("_Texture2", TextureLoader.images[Texture-
Loader.lastHigherKey]);
mat.SetFloat("_Alpha", TextureLoader.lastAlpha);

Graphics.Blit(TextureLoader.images[TextureLoader.lastLowerKey], RenderTex-
ture, mat);
```

Code-Ausschnitt 13: Code um Textur zu erstellen

3.6.5 Hardware Control Panel

Die CO2 Simulation kann neben einem Schieberegler im Orbitarium Control Panel, auch durch einen Drehregler bzw. Joystick auf einem Hardware Control Panel gesteuert werden. Es soll Besuchern des Orbitariums ermöglichend die CO2 Simulation selbst zu steuern.

Abbildung 14: Hardware Control Panel

Dieses Hardware Control Panel wird via USB mit dem PC verbunden. Unity erkennt den Joystick darauf automatisch. Die Achsen und Buttons des Joysticks müssen im Unity Input Manager auf virtuelle Achsen und Buttons gemappt werden. Die virtuellen Achsen bzw. Buttons können beliebige Namen besitzen. Jeder virtuellen Achse bzw. Button können mehrere Joystick Achsen und Buttons zugewiesen werden. Auch Tastatur- oder Maustasten können verwendet werden. Da das Hardware Control Panel eine Achse und zwei Buttons besitzt, die Klimasimulation aber nur eine Achse und einen Button benötigt, wurde neben der «Adjdust Co2» Achse und einem «Reset» Button noch ein Reserve Button definiert. In Abbildung 15 wird das Konfigurationsfenster des Inputmanagers mit der Zuweisung des Joystick Buttons zur virtuellen Achse «Reset Co2» gezeigt.

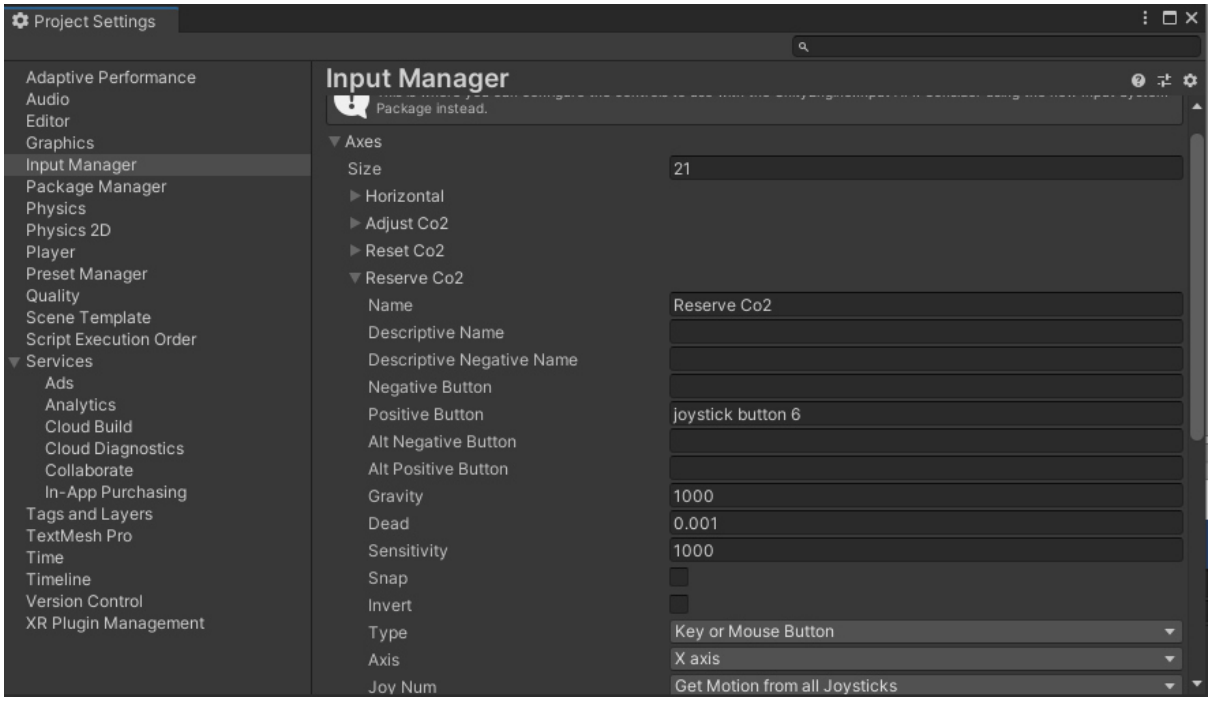


Abbildung 15: Input Manager

Die Joystick Buttons sind von 0 bis 19 durchnummeriert. Um nun herauszufinden, welche Nummern die Buttons des Hardware Control Panels besitzen, werden mit einer For-Schleife die Zustände aller Buttons permanent abgefragt und im Log ausgegeben. Durch Betätigen der Buttons, wird anhand des Outputs die zugehörigen Nummern herausgefunden.

```
for (int i = 0; i < 20; i++)
{
    if (Input.GetKey("joystick 1 button " + i))
    {
        Debug.Log("joystick 1 button " + i);
    }
}
```

Code-Ausschnitt 14: Joystick Buttons abfragen

Das Ergebnis dieses Vorgehens ist in Tabelle 1 niedergeschrieben.

Objekt	Joystick Nummer
C02-Wert / Drehregler	3rd Axis
Reset / Roter Button	Button 7
Reserve / Schwarzer Button	Button 6

Tabelle 1: Zuordnung Joystick Buttons

Sobald das Hardware Control Panel angeschlossen wird, wird der Software Slider deaktiviert. Die Manipulation der Klimasimulation ist dann ausschliesslich via HW-Drehregler möglich. Wird das HW-Control Panel wieder vom Computer getrennt, erhält der Software Slider automatisch seine Funktion zurück.

3.7 URL Quelle

Eine weitere Anforderung, um neuen Input darzustellen, ist es eine Website auf dem Globus zu projizieren, in erster Linie Windy.com.

Um Windy.com auf dem Globus darzustellen ist es naheliegend, diese Website auf dem virtuellen Monitor darzustellen und dann mit der Webcam aufzunehmen. Dadurch kann die Interaktivität immer noch gewährleistet werden. Um die Website auf dem virtuellen Desktop darzustellen, kann der Code der Corona Animation weiterverwendet und mit einem neuen GUI versehen werden. Auf Abbildung 16 ist zu sehen, wie das GUI aussieht, mit welchem eine Website dargestellt werden kann.

Abbildung 16 GUI um Website darzustellen

Um eine beliebige Website zu öffnen, wird ein Input Feld zur Verfügung gestellt. Dieses wird vorgefüllt mit der Windy.com Adresse, um diese schnell öffnen zu können.

3.8 GUI mit Tab-System erweitern

Um die vielen neu Implementierten Features in das GUI einzuarbeiten, muss dieses überarbeitet werden. Die alte Oberfläche ist limitiert darauf, die Corona Animation darzustellen. Um zusätzliche Features darzustellen, wird ein Tab-System verwendet. Abbildung 17 zeigt die aktuelle Lösung. Zurzeit bietet der Unity Asset Store nur sehr wenige, gute Lösungen für solche Grafischen Tab-Systeme. Deshalb wird ein eigenes entwickelt.

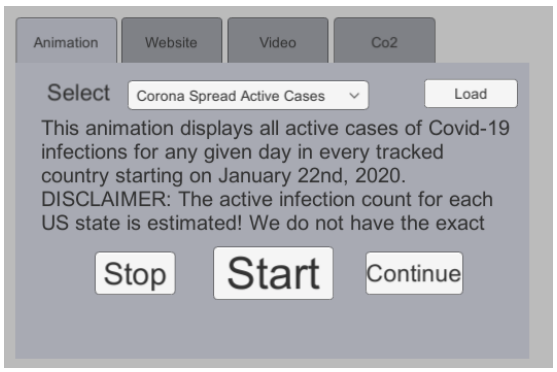


Abbildung 17: Tab-System

Abbildung 18 zeigt, wie das entwickelte System im Editor aussieht. Um das Erweitern der Tabs zu vereinfachen, wurden die Parameter in der Grafischen Oberfläche sichtbar gemacht. Diese bietet zwei Listen an, «Tab Buttons» und «Tabs». In «Tab Buttons» können die Unity Objekte definiert werden, welche die Tabs steuern. In «Tabs» werden die Inhalte der Tabs definiert. Die Zuweisung der beiden Listen erfolgt über den Index der Elemente. Das heisst, das erste Element in der «Tab Buttons» Liste steuert den Inhalt des ersten Elements in der «Tabs» Liste.

Das Aktivieren und Darstellen der Tab Inhalte erfolgt über einen Klick mit der Maus auf das zugehörige Objekt in der «Tab Buttons» Liste. Diese wird dann in der «Active Tab Color» Farbe dargestellt. Zusätzlich werden alle anderen «Tab Buttons» in der «Disabled Tab Color» Farbe dargestellt.

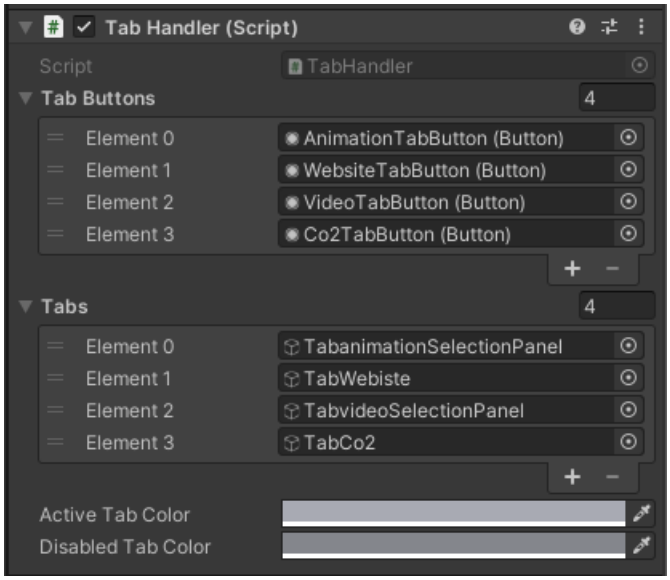


Abbildung 18: Tab Handler Component

4 Diskussion

Im folgenden Kapitel werden die Resultate aus dem vorherigen Kapitel evaluiert und mit der Aufgabenstellung in Kontext gebracht. Ausserdem werden die Resultate mit der vorherigen Implementation anhand einiger Punkte verglichen.

4.1 Performance

Die Performance ist im aktuellen Zustand besser als zuvor. Durch die Analyse und Verbesserung des Codes konnte ein Performancegewinn erzielt werden. Zudem wurde ManyCam als Programm durch OBS Virtualcam ausgetauscht, um eine quadratische Aufnahme zu erlauben. Diese Umstellung verbessert die Performance zusätzlich. Im Vergleich zur alten Lösung gibt es keine Ruckler und merkliche Performanceeinbrüche mehr.

4.2 Unterschiedliche Quellen

Im Orbitarium Control Panel ist es möglich einfach ein Video abzuspielen, mit ein paar Einstellungen kann eine beliebige Website geöffnet und angezeigt werden. Zudem kann eine CO2 Animation abgespielt werden. Im Vergleich zur alten Lösung gibt es mehrere, neue Möglichkeiten, um Inhalte auf dem Globus darzustellen.

4.3 Bedienbarkeit

Durch die Einführung des Tab-Systems und das automatisierte Update der Corona Daten wurde die Bedienbarkeit im Vergleich zur alten Implementation verbessert. Es ist wesentlich einfacher eine neue Quelle auszuwählen und die Corona Daten zu aktualisieren. Ein Regler für Feinjustierung wurde auch implementiert, um die Überschneidung der Projektionen besser steuern zu können.

4.4 Troubleshooting

Auf dem Weg die Ziele zu erreichen, wurden verschiedene Hürden überwunden. Diese waren Zeitintensiv und boten keinen Mehrwert. Um zukünftigen Arbeiten diese Schwierigkeiten zu ersparen, wurden sie mit den entsprechenden Hinweisen zur Lösung zusammengefasst. Diese Tipps sind im Anhang zu finden.

5 Ausblick

Es wurde bereits gezeigt, dass mehrere Möglichkeiten, um Quellen anzuzeigen implementiert sind. Trotzdem gibt es noch einige Punkte, die verbessert werden können.

Das Orbitarium benötigt noch stärkere Projektoren, um auch in nicht komplett dunklen Räumlichkeiten erkennbar zu sein. Es ist bereits in Planung, stärkere Projektoren zu bestellen, sobald das Geld verfügbar ist. Diese müssen dann installiert werden.

Der aktuelle Standort des Orbitariums ist nicht ideal. Es befindet sich in einem eigenen, kleinen, für die Öffentlichkeit unzugänglichen Raum. Zukünftig soll es auf dem Areal der ZHAW, genauer in der Eulachpassage stehen. Der Transport erfordert leider das Zerlegen des Orbitariums. Idealerweise wird das Ersetzen der Projektoren im Rahmen des Umzugs durchgeführt. Ansonsten müsste das Orbitarium zweimal zerlegt werden.

Die Orbitarium-Suite ist noch nicht fertig. Zum einen gibt es im Bereich Robustheit und Benutzerfreundlichkeit Verbesserungspotenzial, zum anderen besteht natürlich immer die Möglichkeit weitere Funktionalität hinzuzufügen.

Auch die Klimasimulation bedarf noch einiger Verbesserung. Sie funktioniert zwar, entspricht jedoch nicht der Realität. Eine Realitätsgetreuere Simulation erfordert aber umfangreiches Wissen in der Meteorologie und kann dadurch nur mit entsprechender Hilfe umgesetzt werden.

6 Glossar

Begriff	Erklärung
Asset	Ein Asset, speziell in Unity, ist ein Objekt oder eine C# Klasse, welche dem Projekt hinzugefügt werden kann und von Unity verwaltet wird.
C#	Eine mit Java und C verwandte Programmiersprache.
GUI	Grafical User Interface. Eine Grafisches Benutzer Schnittstelle.
Hamburgermenü	Button mit 3 horizontalen Balken, um ein Menü hervorzurufen.
IIS	Webserver von Microsoft.
IIS Express	Abgespeckter eingebetteter Webserver für Visual Studio.
Joystick	Physischer Regler, um ein Programm zu steuern.
Jupyter Notebook	Python Notizbuch mit integrierter Codeausführung
Orbitarium	Eine Plexiglaskugel, auf welche mit Projektoren die Oberfläche der Erde projiziert wird.
Orbitarium-Suite	Die Ansammlung an Programmen, welche das Orbitarium betreiben.
REST-Interface	Standardisiertes API.
Shader	Ein Grafikprogramm, um die Pixel zu berechnen, welche auf dem Bildschirm dargestellt werden.
Skript // Skripte	Ein Skript oder Script ist eine Code Klasse, welche in Unity zu Komponenten hinzugefügt werden kann.
Software-Suite	Eine Sammlung von zusammengehörenden Programmen und Dateien.
Textur	Ein Bild oder eine Grafik welche in Unity verwendet wird, um diese im Programm anzuzeigen.
Überblendung	Zwei halbtransparente Bilder werden übereinandergelegt.
Unity	Programm für Spieleentwicklung und weiteres.
Virtualcam	Ein beliebiges Programm, um den Desktop aufzunehmen, welches Input an Unity liefert.

7 Verweise

7.1 Abbildungsverzeichnis

7.2 Tabellenverzeichnis

7.3 Codeverzeichnis

8 Literaturverzeichnis

[1] R. Preuss und N. Schaller, «Neuartiges Orbitarium Darstellungskonzept,» Winterthur, 2020.

[2] «Blue Marble 10000,» [Online]. Available: http://waikiki.zhaw.ch/radar.zhaw.ch/bluemarble10000_en.html. [Zugriff am 03 06 2021].

[3] R. Brem, «Echtzeit Simulation der Auswirkung der Treibhauusgaskonzentration auf das Klima,» ZHAW, Zürich, 2010.

[4] imdevskp, «GitHub,» 08 2020. [Online]. Available: https://github.com/imdevskp/covid_19_jhu_data_web_scrap_and_cleaning.. [Zugriff am 05 06 2021].

[5] «Unity Dokumentation Texture,» Unity, [Online]. Available: https://docs.unity3d.com/Manual/Textures.html. [Zugriff am 08 06 2021].

[6] «Unity Dokumentation Shader,» Unity, [Online]. Available: https://docs.unity3d.com/560/Documentation/Manual/Shaders.html. [Zugriff am 08 06 2021].

[7] «Unity Dokumentation,» Unity, [Online]. Available: https://docs.unity3d.com/ScriptReference/RenderTexture.html. [Zugriff am 05 06 2021].

[8] «Unity Asset Store,» [Online]. Available: https://unity3d.com/quick-guide-to-unity-asset-store#:~:text=The%20Unity%20Asset%20Store%20is,examples%2C%20tutorials%20and%20Editor%20extensions.. [Zugriff am 31 05 2021].

[9] «Simple FIle Browser,» [Online]. Available: https://assetstore.unity.com/packages/tools/gui/runtime-file-browser-113006. [Zugriff am 31 05 2021].

[10] «Unity Dokumentation - Statistiken,» [Online]. Available: https://docs.unity3d.com/Manual/RenderingStatistics.html. [Zugriff am 31 05 2021].

[11] «Unity Dokumentation - Rendering Statistics Window,» Unity, [Online]. Available: https://docs.unity3d.com/Manual/RenderingStatistics.html. [Zugriff am 31 05 2021].

[12] «Unity Dokumentation Profiler,» Unity, [Online]. Available: https://docs.unity3d.com/Manual/Profiler.html. [Zugriff am 31 05 2021].

[13] «Unity Forums Developer Response,» Unity, [Online]. Available: https://forum.unity.com/threads/extremely-poor-webcamtexture-performances-under-android-in-high-resolution.221862/. [Zugriff am 31 05 2021].

[14] «Unity Dokumentation Graphics Blit,» [Online]. Available: https://docs.unity3d.com/ScriptReference/Graphics.Blit.html. [Zugriff am 03 06 2021].

9 Anhang

9.1 Code

Der Code für die Unity Applikation Orbitarium-Control-Panel ist hier abrufbar:
https://github.zhaw.ch/rege/Orbitarium-Control-Panel

Der Code für das Animation-Handling ist hier abrufbar:
https://github.zhaw.ch/rege/Orbitarium-Animation-Service

Das Repo mit den Animationen ist hier abrufbar:
https://github.zhaw.ch/rege/Orbitarium-Animations

9.2 Troubleshooting Tipps

Hier sind einige Erfahrungen, die wir während unserer Bachelorarbeit gemacht haben, niedergeschrieben. Dies soll zukünftigen Arbeiten eine Zeitersparnis bieten.

9.2.1 Äquator nicht waagrecht auf dem Globus.

Sollte der Äquator nicht horizontal, sondern diagonal auf dem Globus liegen, müssen in den Windows Anzeigeeinstellungen Display 2 und 3 vertauscht werden.

9.2.2 Animation Service Codeänderungen nicht im IIS

Der Animation Service muss im Releasemodus gebuildet und anschliessend gepublisht werden.

9.2.3 Activate Display Button funktioniert nicht

Dieser Button funktioniert nur im Build des Projektes. Wenn das Control Panel im Editor gestartet wird, werden die Displays nicht erkannt.

9.2.4 Chrome wird nicht richtig gestartet

Alle Chrome Instanzen müssen geschlossen werden, bevor eine Animation gestartet wird.

Zusätzlich muss der Chromedriver mit dem Chrome zusammenpassen. Am einfachsten ist es Chrome zu updaten und den neusten Chromedriver herunterzuladen. Der Chromedriver muss anschliessend noch ins Verzeichnis «Orbitarium-Control-Panel-Build/Orbitarium Control Panel_Data/Managed» kopiert werden.

9.2.5 Orbitarium Alarm geht nicht aus

Sobald der Globus zu stark pendelt, ertönt eine Sirene. Diese sollte nach einigen Alarmsignalen von selbst verstummen. Ist dies nicht der Fall, ist der Globus in einer Position, die einen Alarmschalter auslöst. Um dies zu beheben, muss der Globus ein bisschen bewegt werden. Dabei ist das Klicken des Schalters zu vernehmen. Anschliessend verstummt der Alarm.

9.2.6 Spacedesk kann keine Verbindung herstellen

Der Spacedesk HTML Client muss sich mit der IP 127.0.0.1 verbinden. Dabei muss der Computer mit einem Netzwerk verbunden sein. Sonst geht es nicht, da keinem Adapter diese IP zugewiesen ist.

9.2.7 Spacedesk Auflösung stimmt nicht

Das Orbitarium benötigt einen Monitor mit einer Auflösung von 2160p x 2160p. Um diese Auflösung einzustellen, wird der Spacedesk Viewer im Browser geöffnet und mit 127.0.0.1 Verbindung hergestellt. Oben Links im Browserfenster können die Einstellungen über das Hamburgermenü aufgerufen werden. Dort kann eine benutzerdefinierte Auflösung eingestellt werden.

9.2.8 Animation Service startet nicht im Visual Studio

Um den Animation Service zu debuggen, wird er im Visual Studio gestartet. Dabei kann es sein, dass der verwendete Port bereits belegt ist. Sollte dies der Fall sein, kann in den Visual Studio Einstellungen ein anderer Port für den IIS Express definiert werden. Zum Beispiel 8080 anstelle 35000.

9.3 Projektmanagement

In den nachfolgenden Kapiteln wird das Projektmanagement beschrieben.

9.3.1 Offizielle Aufgabenstellung

Im den folgenden Unterkapiteln wird die Aufgabenstellung aufgeführt. Der Text wurde aus der originalen Aufgabenstellung übernommen.

9.3.1.1 Unity 3D Visualisierung für Orbitarium

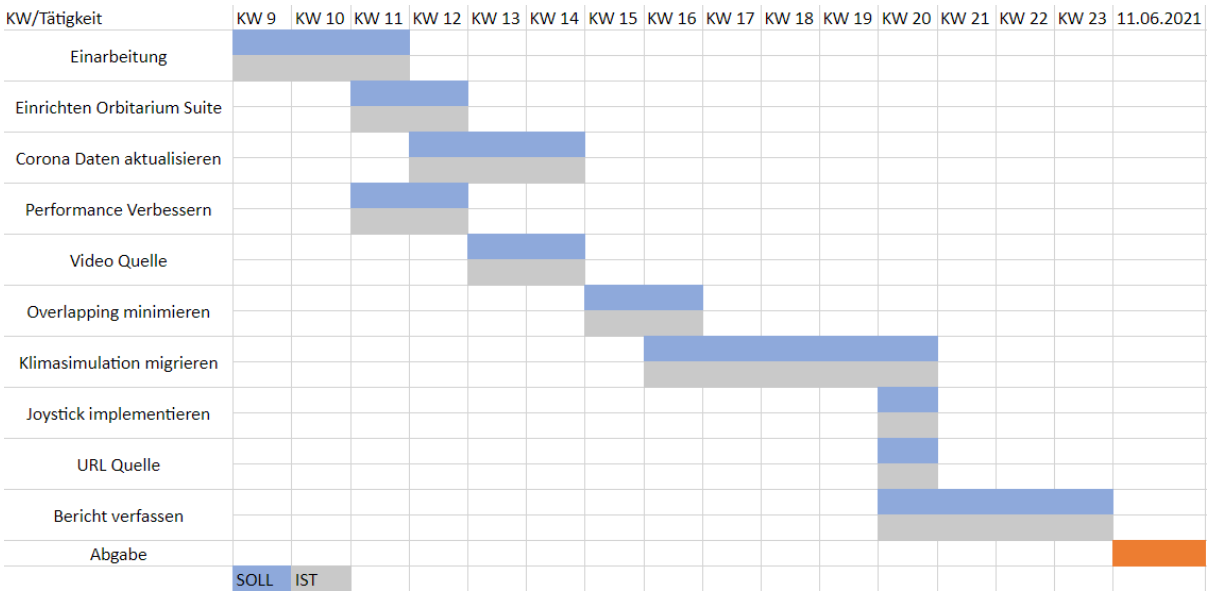
Die ZHAW hat vom Technorama das sogenannte Orbitarium übernommen und plant, für den neuen Campus es an einem gut sichtbaren Ort auszustellen. Es ist ein 1.5m Globus, auf den mittels Beamern dynamische Inhalte projiziert werden können welches zurzeit im Raum direkt neben der T Mensa eingesehen werden kann. Ein Modell der Erde quasi aus der Astronautensicht zu sehen um globale Phänomene zu visualisieren, wird auch im Science on a Sphere Projekt (NOAA) verfolgt, zu dem die ZHAW gute Kontakte hat und schon mehrere Visualisierungen entwickelt hat (globaler Flugverkehr, Corona, Klimaerwärmung). In einer Vorgängerarbeit wurde die in die Jahre gekommene Darstellungssoftware durch eine Unity basierte ersetzt, die auch dynamische Inhalte aus unterschiedlichen Internet Quellen einbinden kann. Dadurch lassen sich neuartige, hochdynamische Visualisierungen erstellen und vielleicht ergeben sich auch Einsatzmöglichkeiten von Augmented Reality via HoloLens. Ob der Fokus mehr auf die Darstellungssoftware oder mehr auf neue Inhalte gelegt wird, das kann mit dem Betreuer nach den Präferenzen der Studierenden noch festgelegt werden.

9.3.1.2 Aufgabenstellung

- Erstellen Sie einen Projektplan
- Bestimmen Sie die grössten Risiken des Projekts
- Recherchieren Sie den aktuellen Stand der Technik
- Entwickeln Sie eine Architektur und die Anwendung
- Testen und verifizieren Sie die Resultate
- Führen Sie jeweils zu den Sitzungen ein (Kurz-)Protokoll
- Betrachten Sie die erzielten Resultate kritisch und überlegen Sie sich mögliche Erweiterungen
- Eine technische Dokumentation (= BA Bericht) und ev. ein einfaches Benutzerhandbuch (im Anhang) soll erstellt werden

9.3.2 Projektplan

Zu Beginn der Bachelorarbeit wurde ein Projektplan erstellt, um die geplanten Tätigkeiten zeitlich einzuteilen.



9.3.3 Besprechungsprotokolle

Während der gesamten Projektdauer wurden Wöchentliche Statusmeetings durchgeführt. Jedes dieser Meetings wurde in einem Dokument festgehalten. Diese befinden sich auf den nächsten Seiten im Anhang.

Sitzungsprotokoll

Sitzungsprotokoll zur BA Orbitarium der Semesterwoche 1

Dozent:	Karl Rege	**Datum:**	23.02.2021
Studierende:	Tim Krenz, krenztim Kevin Thalmann, thalmkev		
Abwesende:	-		

Erledigte Arbeiten

1. Einarbeitung in die Materie
2. Erstellung des Protokolls und GIT Repository

Offene Fragen

1. Was sind die Bewertungskriterien?
2. Was sind die Ansprüche an den Bericht / die Präsentation?
3. Bekommen wir noch Zugriff auf den Source Code via Git?
    a. Ist erledigt
4. Kann die Applikation auch getestet werden ohne den Orbitarium‘s PC (von zuhause zum Beispiel)?
    a. Ein Simulator ist vorhanden

Neuer Input / Erkenntnisse

1. SpaceDesk ist aus der Beta gekommen. Die Version Läuft ohne Anpassungen nicht mehr
    a. SpaceDesk war veraltet -> update
    b. ChromeDriver war veraltet -> update
2. Taskbar war auf Bildschirm 2 und 3 eingeblendet, was dazu geführt hat, dass das Bild verzerrt war und die Taskbar immer angezeigt wurde -> Taskbar auf anderen Bildschirmen ausschalten

Ausblick

3. Backup Orbitarium PC
4. ~~SpaceDesk Workaround finden oder Software alternative finden.~~
    a. Sicherstellen SpaceDesk und ChromeDriver funktionieren in Zukunft.
5. Wlan Dongel kaufen für Orbitarium PC
6. Tests / Qualität verbessern
7. Fehlermeldungen einbauen
8. Dokumentation anpassen
9. Animation flüssiger machen

Sitzungsprotokoll

Sitzungsprotokoll zur BA Orbitarium der Semesterwoche 1

Dozent:	Karl Rege	**Datum:**	02.03.2021
Studierende:	Tim Krenz, krenztim Kevin Thalmann, thalmkev		
Abwesende:	Tim Krenz, krenztim		

Erledigte Arbeiten

- Unity installieren
- Einarbeitung
- Erarbeiten der Aufträge

Offene Fragen

- Was sind die Bewertungskriterien?
    - Auf der Website von Karl Rege verfügbar.

Neuer Input / Erkenntnisse

-

Ausblick

Arbeiten nächste Woche

- Orbitarium Pc backup
- Chrome Auto-Update deaktivieren
- Wlan Dongel kaufen
- Animation flüssiger machen

Arbeiten für später

- Dokumentaiton anpassen
- Hauptsicht „unverzerrte Sicht“ wieder einblenden
- Neue Beamer kalibrieren
- Overlaping minimieren (variable gestalten
- Corona Datensatz erweitern SOS

Sitzungsprotokoll

Sitzungsprotokoll zur BA Orbitarium der Semesterwoche 3

Dozent:	Karl Rege	**Datum:**	09.03.2021
Studierende:	Tim Krenz, krenztim Kevin Thalmann, thalmkev		
Abwesende:			

Erledigte Arbeiten

- Orbitarium Pc backup vorbereitet
- Chrome Auto-Update deaktivieren vorbereitet
- Wlan Dongel gekauft
- Animation flüssiger machen -> Problem erkannt

Offene Fragen

Neuer Input / Erkenntnisse

- Video Streams in Unity warden in anderen BA's gebraucht, evtl kommunikation.

Ausblick

Arbeiten nächste Woche

- Orbitarium Pc backup
- Chrome Auto-Update deaktivieren
- Wlan Dongel anschliessen
- Animation flüssiger machen -> Lösung suchen

Arbeiten für später

- Dokumentaiton anpassen
- Hauptsicht „unverzerrte Sicht" wieder einblenden
- Neue Beamer kalibrieren
- Overlaping minimieren (variable gestalten
- Corona Datensatz erweitern SOS

Sitzungsprotokoll

Sitzungsprotokoll zur BA Orbitarium der Semesterwoche 4

Dozent:	Karl Rege	**Datum:**	16.03.2021
Studierende:	Tim Krenz, krenztim Kevin Thalmann, thalmkev		
Abwesende:			

Erledigte Arbeiten

- Orbitarium Pc backup begonnen
- Chrome Auto-Update deaktiviert
- Wlan Dongel angeschlossen
- Animation flüssiger machen -> Code refactoring mehr fps gebracht. Mögliche performance increase mit Auflösung verändern

Offene Fragen

Neuer Input / Erkenntnisse

Ausblick

Arbeiten nächste Woche

- Orbitarium Pc backup weitertreiben
- Corona Datensatz erweitern auf aktuellen Stand
- Video auf Globus projiezieren
    - Neue Datenquelle
    - Video zu Globus Darstellung
    - Auswahl in Unity Applikation

Arbeiten für später

- Dokumentaiton anpassen
- Hauptsicht „unverzerrte Sicht“ wieder einblenden
- Neue Beamer kalibrieren
- Overlaping minimieren (variable gestalten

Sitzungsprotokoll

Sitzungsprotokoll zur BA Orbitarium der Semesterwoche 5

Dozent:	Karl Rege	**Datum:**	23.03.2021
Studierende:	Tim Krenz, krenztim Kevin Thalmann, thalmkev		
Abwesende:			

Erledigte Arbeiten

- Corona Datensatz erweitert auf aktuellen Stand
- Video auf Globus projizieren – Fast fertig

Offene Fragen

Neuer Input / Erkenntnisse

Ausblick

Arbeiten nächste Woche

- Orbitarium Pc backup weitertreiben
- Corona Datensatz Animation bug fixen
- Video auf Globus projizieren fertig stellen

Arbeiten für später

- Dokumentation anpassen
- Hauptsicht „unverzerrte Sicht“ wieder einblenden
- Neue Beamer kalibrieren
- Overlaping minimieren (variable gestalten

Sitzungsprotokoll

Sitzungsprotokoll zur BA Orbitarium der Semesterwoche 6

Dozent:	Karl Rege	**Datum:**	30.03.2021
Studierende:	Tim Krenz, krenztim Kevin Thalmann, thalmkev		
Abwesende:			

Erledigte Arbeiten

- Corona Datensatz Animation bug fixen
- Video auf Globus projizieren fertig stellen
- Orbitarium Pc backup erledigt

Offene Fragen

Neuer Input / Erkenntnisse

Ausblick

Arbeiten nächste Woche

- Corona Datensatz Film „Active Cases”
- Hauptsicht „unverzerrte Sicht“ wieder einblenden
- Video Auflösung anschauen
- Overlaping minimieren (variable gestalten)
- CO2 Datensatz

Arbeiten für später

- Dokumentation anpassen
- Neue Beamer kalibrieren

Sitzungsprotokoll

Sitzungsprotokoll zur BA Orbitarium der Semesterwoche 7

Dozent:	Karl Rege	**Datum:**	06.04.2021
Studierende:	Tim Krenz, krenztim Kevin Thalmann, thalmkev		
Abwesende:	Karl Rege		

Erledigte Arbeiten

- Corona Datensatz Film „Active Cases" erstellt
- Hauptsicht „unverzerrte Sicht" wieder einblenden angeschaut
- Video Auflösung angeschaut
- Overlaping minimieren (variable gestalten) angefangen
- CO2 Datensatz Kontakt aufgenommen

Offene Fragen

Neuer Input / Erkenntnisse

- Kontakt für Co2 Datensatz meldet sich nicht

Ausblick

Arbeiten nächste Woche

- Corona Datensatz Film „Active Cases"
- Hauptsicht „unverzerrte Sicht" wieder einblenden
- Video Auflösung anschauen
- Overlaping minimieren (variable gestalten)
- CO2 Datensatz

Arbeiten für später

- Dokumentation anpassen
- Neue Beamer kalibrieren

Sitzungsprotokoll

Sitzungsprotokoll zur BA Orbitarium der Semesterwoche 8

Dozent:	Karl Rege	**Datum:**	13.04.2021
Studierende:	Tim Krenz, krenztim Kevin Thalmann, thalmkev		
Abwesende:			

Erledigte Arbeiten

- Corona Datensatz Film „Active Cases" erstellt
- Hauptsicht „unverzerrte Sicht" eingebaut
- Video Auflösung angeschaut -> macht keine Probleme
- Overlaping minimieren (variable gestalten) -> muss im Orbitarium getestet werden
- CO2 Datensatz Bachelorarbeit angeschaut

Offene Fragen

Neuer Input / Erkenntnisse

- Kontakt für Co2 Datensatz: Doris Folini
- 32 bit – für Programm benötigt
- Bilder nicht neu generieren -> nur welches Bild passt gut zu welcher Temperatur
- Model auf akzeptablen Stand bringen (Wissenschaftlich fundiert)

Ausblick

Arbeiten nächste Woche

- Neues Programm, um Bildschirm aufzunehmen (OBS)
- Applikation GUI besser gestalten
- Overlaping im Orbitarium testen
- Java Applikation zum Laufen bringen (co2 letzten Mail)
- Model reinlesen

Arbeiten für später

- Dokumentation anpassen
- Neue Beamer kalibrieren

Sitzungsprotokoll

Sitzungsprotokoll zur BA Orbitarium der Semesterwoche 9

Dozent:	Karl Rege	**Datum:**	20.04.2021
Studierende:	Tim Krenz, krenztim Kevin Thalmann, thalmkev		
Abwesende:			

Erledigte Arbeiten

- Neues Programm, um Bildschirm aufzunehmen (OBS)
- Java Applikation zum Laufen bringen (co2 letzten Mail)
- Model reinlesen

Offene Fragen

Neuer Input / Erkenntnisse

Gegeben Co2 Input, Output Modell für Verteilung Wasser / Eis.
Energiehaushalt.

Modell beschränkt auf:

Sonne, Erde, Wolke, Berge.

Albedo weiss = reflektiert

Regional aufgelöstes Modell

Modell von Masterarbeit nehmen und numerisch belegen

Ausblick

Arbeiten nächste Woche

- Neues Programm, um Bildschirm aufzunehmen (OBS)
- Applikation GUI besser gestalten
- Overlaping im Orbitarium testen
- Überlbenden von Bildern
- Modell erarbeiten

Arbeiten für später

- Dokumentation anpassen
- Neue Beamer kalibrieren

Sitzungsprotokoll

Sitzungsprotokoll zur BA Orbitarium der Semesterwoche 10

Dozent:	Karl Rege	**Datum:**	27.04.2021
Studierende:	Tim Krenz, krenztim Kevin Thalmann, thalmkev		
Abwesende:			

Erledigte Arbeiten

- Model erarbeiten
- GUI anpassem

Offene Fragen

Neuer Input / Erkenntnisse

Code von Rege in Unity darstellen

Brem Modell verifizieren

Ausblick

Arbeiten nächste Woche

- Neues Programm, um Bildschirm aufzunehmen (OBS)
- Overlaping im Orbitarium testen
- Überlbenden von Bildern
- Applikation von Java in C# übernehmen
- Modell von Brem verifizieren

Arbeiten für später

- Dokumentation anpassen
- Neue Beamer kalibrieren

Sitzungsprotokoll

Sitzungsprotokoll zur BA Orbitarium der Semesterwoche 11

Dozent:	Karl Rege	**Datum:**	04.05.2021
Studierende:	Tim Krenz, krenztim Kevin Thalmann, thalmkev		
Abwesende:			

Erledigte Arbeiten

- Termin mit Breme verhandelt
- OBS auf Orbitarium installiert
- Überblenden von Bildern
- Animation Co2 in Unity ipmlementiert

Offene Fragen

Neuer Input / Erkenntnisse

11.06.2021 Bericht fertig schreiben
2-3 Wochen nach Abgabe Präsentation

Ausblick

Arbeiten nächste Woche

- Automatisches Update für Corona Daten
- Windy.com reinnehmen

Arbeiten für später

- Dokumentation anpassen
- Neue Beamer kalibrieren

Sitzungsprotokoll

Sitzungsprotokoll zur BA Orbitarium der Semesterwoche 12

Dozent:	Karl Rege	**Datum:**	11.05.2021
Studierende:	Tim Krenz, krenztim Kevin Thalmann, thalmkev		
Abwesende:			

Erledigte Arbeiten

- Animation Co2 in Unity Bugfixes
- Automatisches Update für Corona Daten

Offene Fragen

Neuer Input / Erkenntnisse

11.06.2021 Bericht fertig schreiben
2-3 Wochen nach Abgabe Präsentation

Ausblick

Arbeiten nächste Woche

- Automatisches Update für Corona Daten Dokumentieren
- Windy.com reinnehmen

Arbeiten für später

- Dokumentation anpassen
- Neue Beamer kalibrieren

Sitzungsprotokoll

Sitzungsprotokoll zur BA Orbitarium der Semesterwoche 13

Dozent:	Karl Rege	**Datum:**	18.05.2021
Studierende:	Tim Krenz, krenztim Kevin Thalmann, thalmkev		
Abwesende:			

Erledigte Arbeiten

- Animation Co2 in Unity Bugfixes
- Automatisches Update für Corona Daten
- Windy.com reinnehmen

Offene Fragen

Neuer Input / Erkenntnisse

11.06.2021 Bericht fertig schreiben
2-3 Wochen nach Abgabe Präsentation

Ausblick

Arbeiten nächste Woche

- Automatisches Update für Corona Daten Dokumentieren
- Umrechnung Physikalischer Regler

Arbeiten für später